FUNCTIONALITY

For Measuring Functionality of software we examine 3 Approaches

1) Albrecht's function point analysis
2) De Marco's specification weight
3) COCOMO 2.0 approach to object points

1) **Albrecht's approach for effort estimation**

Function points are intended to measure the amount of functionality in a system as described by a specification
To compute FP, first compute unadjusted function point count (UFC)

To compute UFC, we have to determine the number of items of following types :

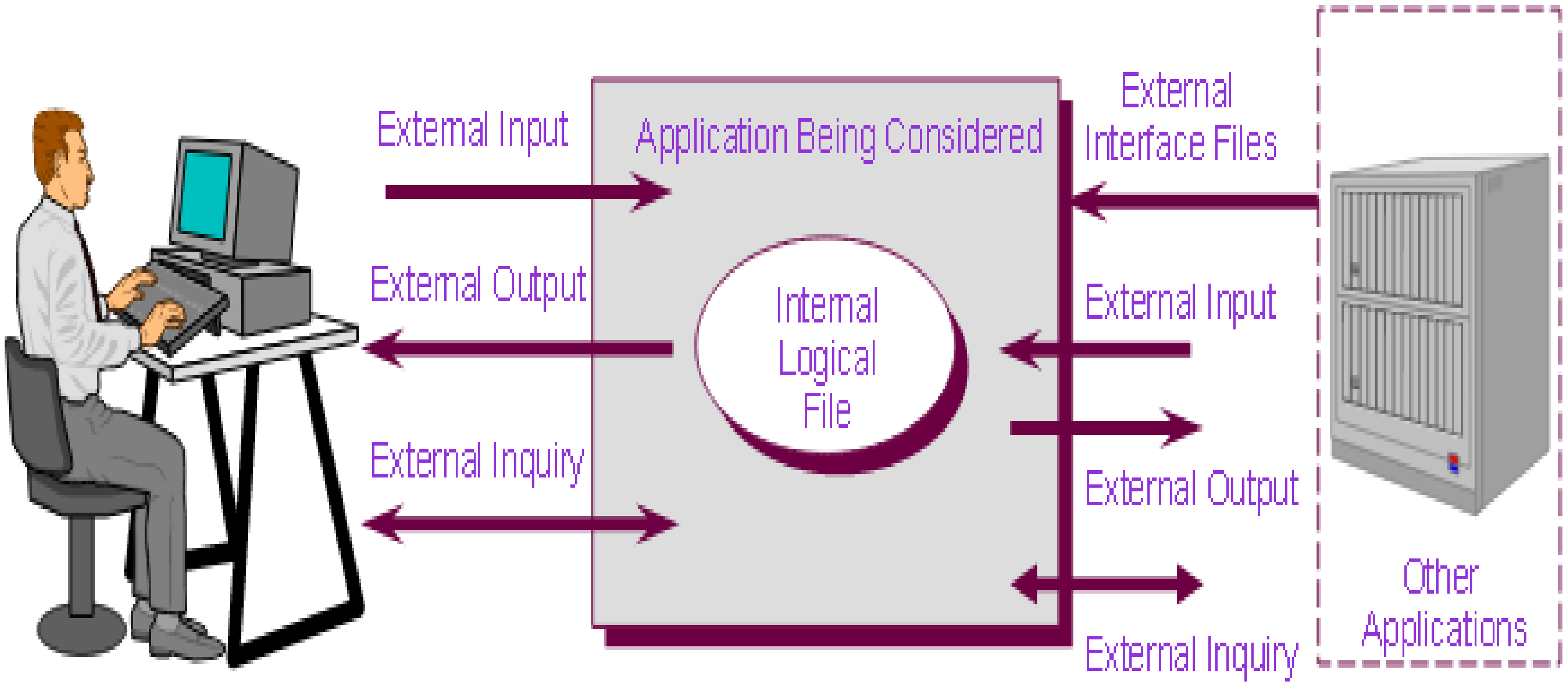
1) **External Inputs** Those items provided by user that describe distinct application oriented data (file names & menu selections) These items do not include inquiries
2) **External outputs** Reports & Messages on screen
3) **External Inquiries** Interactive inputs requiring a response
4) **External files** Machine-readable interfaces to other systems
5) **Internal files** Logical master files in the system

Weighting Factor

Item	Simple	Average	Complex
Ext inputs	3	4	6
Ext outputs	4	5	7
Ext inquiries	3	4	6
Ext files	7	10	15
Int files	5	7	10

FP-Based Estimation

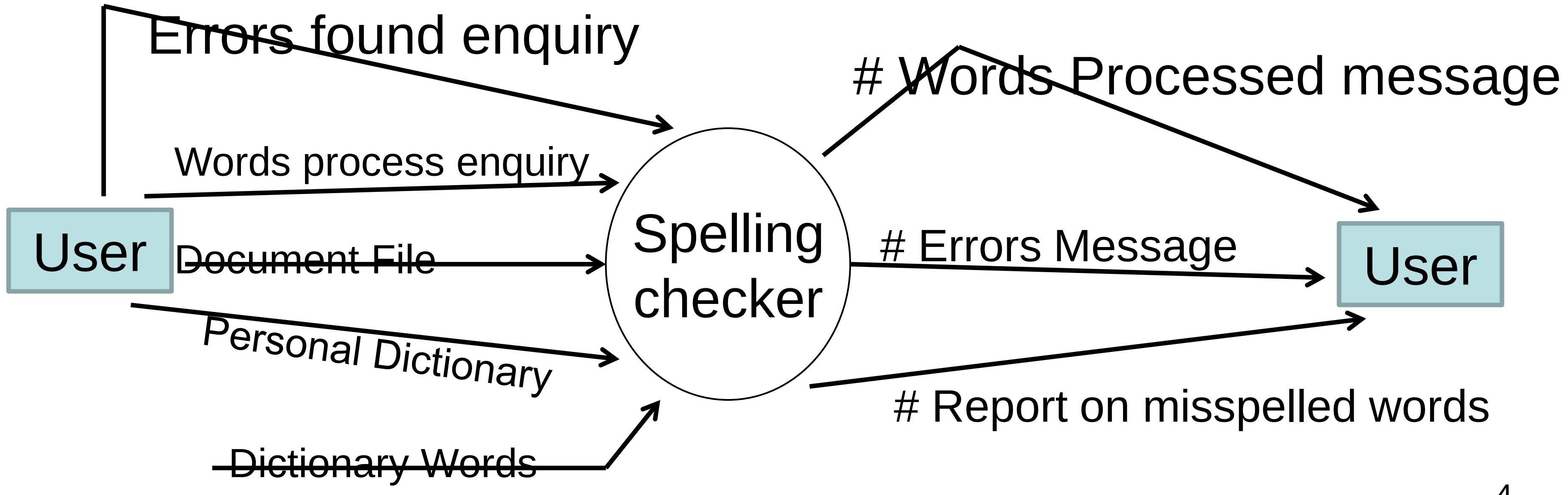
FPA functional units



Example : Computing basic function point components from specification

Spell checker specification

The checker accepts as input a document file and an optional Personal dictionary file. The checker lists all words not contained in either of these files. The user can query the number of words processed and the number of spelling errors found at any stage during processing



A=2

B=3

C=2

D=2

E=1

There are 2 external inputs – document file and personal
A = 2 dictionary file

Three external outputs B = 3 misspelled word report
number of words processed message
number of errors message

C = 2 External inquiries Words processed
Errors so far

D = 2 External files document file & Personal dictionary

E = 1 Internal file Dictionary file

Each item is assigned a subjective complexity as simple, Average or complex.

Here we assume the complexity for each item is average

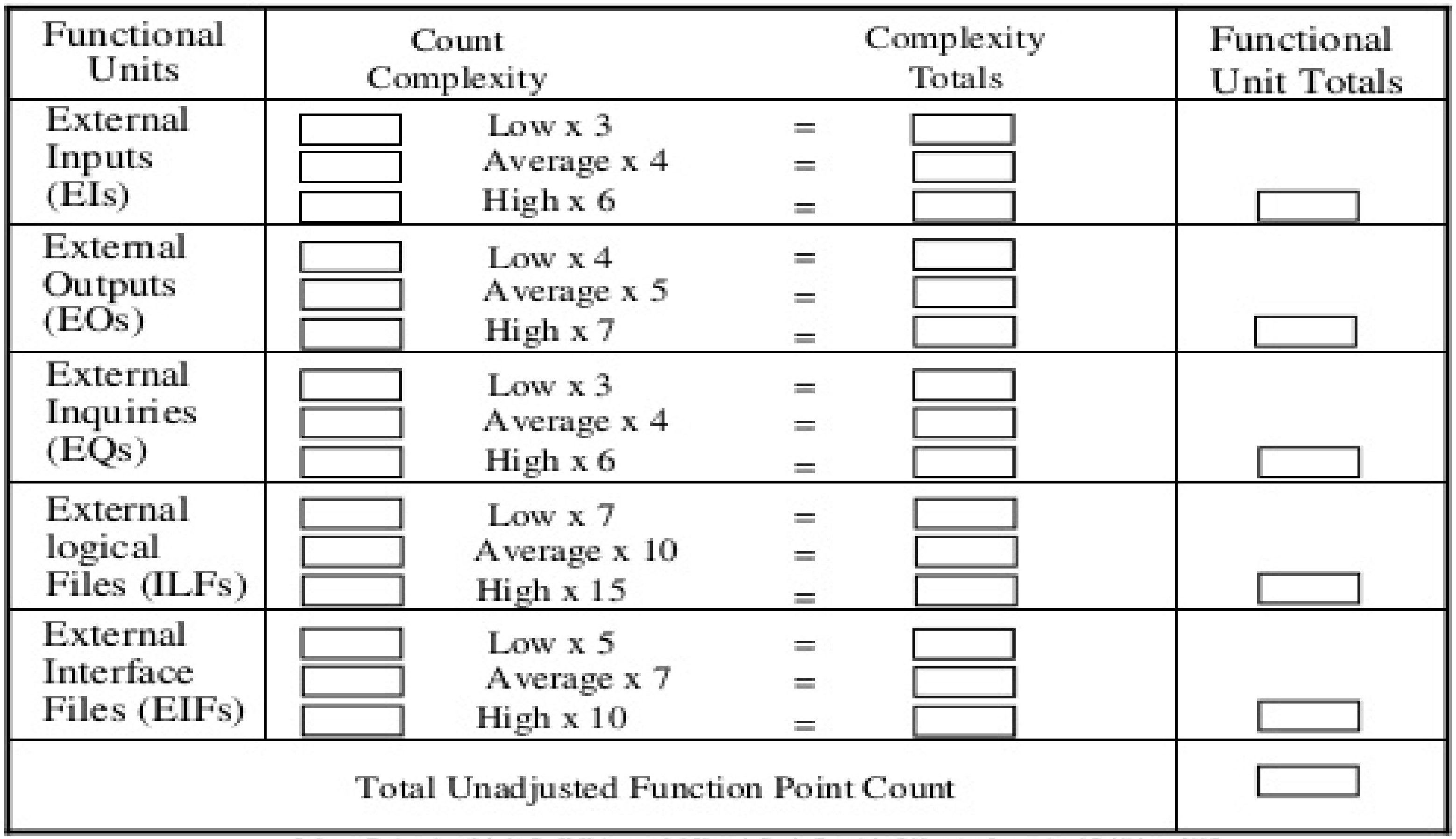
UFC= 4A + 5B + 4C + 10D + 7 E

UFC = 4(2) + 5(3) + 4(2) + 10(2) + 7(1) = 58

FP-Based Estimation

- ***Unadjusted Function Points (UFP)***

UFP Calculation table



Functional Units	Count Complexity			Complexity Totals	Functional Unit Totals
External Inputs (EIs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External Outputs (EOs)		Low x 4	=		
		Average x 5	=		
		High x 7	=		
External Inquiries (EQs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External logical Files (ILFs)		Low x 7	=		
		Average x 10	=		
		High x 15	=		
External Interface Files (EIFs)		Low x 5	=		
		Average x 7	=		
		High x 10	=		
Total Unadjusted Function Point Count					

If for example, dictionary file and misspelled word report Are complex then UFC will change as :

UFC = (4(1)+6(1)) + (5(2)+7(1)) + 4(2) + (10(1)+15(1)) + 10(1) = 70

To compute function points, we calculate adjusted function Point count FP. FP = UFC X Technical complexity factor (TCF)

TCF involves 14 contributing factors

Each factor in table is rated from 0 to 5, where 0 means that factor is irrelevant and 3 means average, 5 means it is essential to systems being built.

$$TCF = 0.65 + 0.01 \sum_{i=1}^{14} F_i$$

F1 : Reliable backup and Recovery
F2 : Data Communications
F3 : Distributed functions
F4 : Performance
F5 : Heavily used configuration
F6 : Online Data entry
F7 : Operational Ease
F8 : Online Update
F9 : Complex interface
F10 : Complex processing
F11 : Reusability
F12 : Installation ease
F13 : Multiple Sites
F14 : Facilitate change

For given example F3, F5, F9, F11, F12, F13 are 0
F1, F2, F6, F7, F8 ANF F14 ARE 3 and that
F4 and F10 are 5

Hence TCF = 0.65 + 0.01 (18 + 10) = 0.93
UFC = 70 (Already calculated)
FP = 70 x 0.93 = 65 Function Points

Suppose our historical database or project measurements
reveals that it takes a developer an average of 2 person
Days of effort to implement a function point, then we may
Estimate the effort needed to complete the spelling
Checker as 65 x 2 = 130 person days
or 4.3 work months.

FP-Based Estimation – An Example (cont..)

- Consider a project with the following functional units:

Number of user inputs	= 50
Number of user outputs	= 40
Number of user enquiries	= 35
Number of user files	= 06
Number of external interface	= 04

Assume all complexity adjustment factors and weighting factors are average. Compute function points for the project. Suppose that program needs 70 LOC per FP. Find out the size of complete project

$$UFP = 50 * 4 + 40 * 5 + 35 * 4 + 6 * 10 + 4 * 7$$
$$= 200+200+140+60+28 = 628$$
$$CAF = (0.65 + \sum F_i)$$
$$=(0.65 + 0.01\ (14 * 3)) = 1.07$$
$$FP = UFP * CAF$$
$$=628 * 1.07 = 672$$
$$Size = FP * (LOC\ per\ FP) = 672 * 70 = 47040\ LOC$$

Exercises:

2. Consider a project with the following parameters
   1. **External Inputs:** 10 with low complexity, 15 with average complexity, 17 with high complexity
   2. **External Outputs:** 6 with low complexity, 13 with high complexity
   3. **External Inquiries:** 3 with low complexity, 4 with average complexity, 2 with high complexity
   4. **Internal logical files:** 2 with average complexity, 1 with high complexity
   5. **External Interface files:** 9 with low complexity

   In addition to above, system requires

   - Significant data communication
   - Performance is very critical
   - Designed code may be moderately reusable
   - System is not designed for multiple installations in different organizations

   Other complexity adjustment factors are treated as average. Compute the function points for the project

Exercises:

3. **An application has the following; 10 low external inputs, 12 high external outputs, 20 low internal logical files, 15 high external interface files, 12 average external inquiries and a value of complexity adjustment factor 1.10. What are the unadjusted and adjusted function point counts?**

Function points are also used in other ways as a size Measure

1) We can express defect density in terms of defects per function point
2) They are also used in contracts, to report progress and to define payment
   50% to 60% software contracts in Netherlands have their
Costs tied to function-points specification.
3) Price per function point may be fixed during software contracts
4) We track project completion by reporting number of function points specified, designed, coded and tested

COCOMO 2.0 approach (a model for predicting effort)

Object points was selected for size input

To compute Object Points, an initial size measure is generated by counting number of screens, etc.
Next each object is classified as simple, medium and Difficult as per given guidelines
Object point complexity levels
For Screens

Number of views Contained	Number and source of data tables		
	Total < 4 (<2 server, <2 client)	Total < 8 (2-3 server, (3-5 client)	Total 8+ (>3 server >5 client)
< 3	simple	simple	medium
3-7	simple	medium	difficult
8+	medium	difficult	difficult

For Reports

Number of sections Contained	Number and source of data tables: Total < 4 (<2 server, <2 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server > 5 client)
0 or 1	simple	simple	medium
2 or 3	simple	medium	difficult
4+	medium	difficult	difficult

The number in each cell is weighted according to the given Table

Object Type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
4GL component	--	--	10

(4GLs such as SAS, SPSS, Stata, ORACLE, etc)

The weight reflect the relative effort required to implement an Instance of that complexity level.
Then weighted instances are summed to yield a single Object-point number.
Then Re-use is taken into account.
Assuming that r% of objects will be reused from previous Projects, the number of new object points is calculated to be

$$\text{New Object points} = (\text{object points}) \times (100 - r) / 100$$

To use this number for effort estimation, COCOMO 2.0 Determines a productivity rate (i.e new object points per Person month) from a table based on developer experience and capability.

Example :

Suppose 840 object points are computed from a system Specification and 20% can be supplied by existing Components the
NOP= 840 x (100 – 20) / 100 = 672 object points

<u>Object Points effort conversion table</u>

Developer's experience	**Very Low**	**Low**	**Nominal**	**High**	**Very High**
Productivity Factor	4	7	13	25	50

In our example, productivity is nominal
Effort = 672 / 13 = 52 Months