FUNCTION POINT METHODOLOGY:

Purpose of function point analysis:

- It determines the size of the software application or project in order to estimate the time and effort required to complete it.
- It measures productivity in function points per staff or calendar month.
- It estimates the cost of changing systems.
- It evaluates support requirements.
- It monitors outsourcing agreements.
- It normalises the comparison of software modules.

Determining the function point count:

In order to accomplish the purpose of the function point methodology above, *Garmus D, & Herron D,[2]*; **suggest the following steps to size function points:**

1. Determine the type of function point count. Is it a new development, an enhancement, or an existing application that has to be counted?
2. Identify the counting scope and application boundaries.
3. ***Identify all:***
   - *3.1* IFPUG - Data functions (internal logical files and external interface files) and their complexity;
   - *3.2* Mark II - Logical Transactions (LT).

4. ***Identify all:***
   - 4.1 IFPUG - External user types also referred to as 'transactional functions' (external input, external output, and external inquiries) and their complexity.
   - 4.2 Mark II - Functional size of each input, output & process transaction associated with each LT.
5. Determine the unadjusted function point count.
6. Determine the value adjustment factor.
7. Calculate the value adjustment factor.

These seven steps can be loosely grouped into two distinct phases: steps 1 to 5 focus on the counting of a project or application's function points, whereas steps 6 and 7 adjust the value of these function points. We will limit the discussion below to steps 2 to 5 in order to illustrate the application of both methodologies.

Application of steps 2 to 5 in the IFPUG function point methodology:

Step 2 - Determine the project or application boundary:

The project or application boundary identifies the border between the application being measured, the users and the external applications. In IFPUG, the boundary is based on the scope of the project as seen by the user. *Garmus & Herron [2]* recommend that boundaries should not be established at the individual software program level, but at a higher application level such as system level.

Steps 3 and 4

The IFPUG 4.0 methodology analyses software requirements or user functionality relative to five component types, two of which are data-related and the remaining three are transaction-related.

Step 3 - Identify the data-related components:

- IFPUG makes use of *two data-related components*. The classification of the data into Internal Logical files (ILF) or External Interface files (EIF) depends on whether the maintenance of the files (or database) falls within the project or application boundary or outside the boundary.
- From the above it is clear that EIF and ILF are mutually exclusive, contrary to the information contained in the textbook. Refer to page 114. A file can therefore not be classified as both an ILF and an EIF.

Internal Logical File types (ILF) -

- An internal logical file type (ILF) is a group of logically related data elements maintained *within* the boundary of the application. *'The primary intent of the ILF is to hold data maintained through one or more elementary processes of the application being counted.'* [2]
- Each file or database can consist of subgroups referred to as record types. If different processing logic is required for each record type, the record types are counted separately.
- The complexity of the file is determined by assigning each file a functional complexity of low, average or high; based on the number of data element types and record element types associated with the ILF according to Table 5.3 on page 115 of the textbook.

External Interface File types (EIF) -

- IFPUG defines an external interface file type (EIF) as a group of logically related data elements referenced by the application but maintained within the boundary of a different application. The purpose of the EIF is to *'hold data referenced through one or more elementary processes within the boundary of the application counted. An EIF for an application must be an ILF in another application'.* [2]
- The complexity of the EIF file is determined by assigning each file a functional complexity of low, average or high; based on the number of data element types and record element types associated with that EIF according to Table 5.3 on page 115 of the textbook.

Step 4 - Determine the transactional functions:

External Input transactions (EI) -

- An EI is a user-generated input transaction that:
  - Enters from outside the project boundary.
  - Processes data or systems control information.
  - Maintains one or more ILFs and/or changes the behaviour of the system.
- The following table is used to determine the complexity rating of the EI based on the number of data types and number of file types (ILF and EIF) accessed by the EI.

External Output transactions (EO) -

- An EO is an:
  - *'elementary process of the application that generates data or control information that exits the boundary of the application.*
  - *The primary intent of an EO is to present information to a user through the retrieval of data or control information from an ILF or EIF.*
  - *The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILF's, or alter the behaviour of the system.'* [2]
- The number of data and file types referenced determine the complexity of an EO. Refer to Table 5.5 of textbook (see below).

External Inquiry (EQ) -

- An External Inquiry (EQ):
  - Consists of an input/output pair that crosses the boundary.
  - *'Results in retrieval of data or control information that is sent outside the application boundary.*
  - The primary intent is to present information to a user through the retrieval of data or control information from an ILF or EIF.
  - The processing logic contains no mathematical formulas or calculations and creates no derived data. No ILF is maintained during processing, and the behaviour of the application is not altered.' [2]
- The classification of transactions as EI and EO or EQ is mutually exclusive.
- The complexity of an EQ is determined by establishing a complexity rating for both the input and output part of the enquiry separately as per Table 5.4 and 5.5 (see below). The higher complexity rating is chosen as the complexity score for the EQ.

TABLE 5.4 IFPUG EXTERNAL INPUT COMPLEXITY

Number of file types accessed	***Number of data types accessed.***		
	<5	***5 to 15***	***>15***
0 or 1	Low	Low	Average
2	Low	Average	High
>2	Average	High	High

TABLE 5.5 IFPUG EXTERNAL OUTPUT COMPLEXITY

Number of file types accessed	***Number of data types accessed.***		
	<6	***6 to 19***	***>19***
0 or 1	Low	Low	Average
2 or 3	Low	Average	High
>3	Average	High	High

Step 5 - Determine the unadjusted function point count:

- Based on the complexity rating derived for the five component types, Table 5.2 on page 115 of the textbook is referenced to determine the function point score for each component or external user type.
- The function points are summed to give the total IFPUG unadjusted function point count for the project or application being sized.

Application of steps 2 to 5 in the Mark II function point methodology

The Mark II (Mk II) methodology assumes a model of software where all user requirements are expressed in terms of 'Logical Transactions' (LT). Each LT comprises an input, some processing and an output component. An LT is triggered by an event outside the systems boundary, such as a request for information.

Step 2 - Determine the project or application boundary:

- As with IFPUG, the boundary of the project or application must be established. The boundary determines which logical transactions are to be included in the count, as well as the interfaces required to link to LT outside the application boundary.

Step 3 - Identify the Logical Transactions (LT):

- Logical transactions are the smallest complete units of information processing that are meaningful to the end user.
- As an example, 'Maintain Customer Details' is not the lowest-level business process, but implies a series of LT such as 'View' an existing item, if 'Not Found' add a new item, 'Change' the viewed item or 'Delete' it, etc.
- Each LT consists of three components: input across an application boundary, processing involving stored data within the boundary and output back across the boundary.
- Each LT must be self-contained and leave the application in a consistent state.

Step 4 - Identify and determine the functional size of the three components of each LT:

The processing component of the LT -

- The processing component of each LT is analysed by:
  - referencing its manipulation (i.e. create, read, update, list or delete),
  - stored data expressed logically as primary and primary entity subtypes in third normal form.
- A primary entity type is: *'One of the main entity-types which has the attributes that the application has been designed to process and/or store.'* [5] Example: An Employee would be an entity type in a Personnel system.
- The primary entity type may consist of primary entity subtypes. It is a subdivision of an entity type which has all the attributes and relationships of its parent entity type, and may have additional unique attributes and relationships i.e. the Employee entity may consist of Personal, Work Experience and Job History detail with different processing rules. Subtypes are counted separately, depending on the processing logic used. (Subtypes can roughly be compared to IFPUG's record types.)
- The processing component of a LT is sized by counting the number of primary entity or sub-entity types that are referenced by the LT.

Input and output components of LT -

- The input element consists of the retrieval and validation of incoming data, such as a request for information or data input by the user.
- The output element consists of formatting and presentation of information to the user.
- The input and output components are sized by counting the number of data element types crossing the application boundary (inside and outside).
- A data element type is a uniquely processed item of information that is indivisible for the purposes of a LT being sized.

Step 5 - Determine the unadjusted function point count:

- The functional size of a LT is the weighted sum of the input, processing, and output components of the LT.
- The industry standard weights are as follows: Wi=0.58, We=1.66 and Wo=0.26.
- The sum of all LT determines the Mark II FP count.

A comparison between Mark II and IFPUG function point methodologies

General:

1. IFPUG has a user base exceeding 1200 member companies in the USA. Mark II is mainly used in Europe and the UK. Although Mark II had some advantages over the initial Albrecht FPA, IFPUG has been enhanced to such an extent that the advantages have largely disappeared.

2. One aspect in which Mark II offers more refinements is in the area of Technical Complexity Adjustment factors. Both techniques use similar approaches to adjust for qualitative requirements. They also use a list of non-functional system characteristics which are evaluated on a scale from 0 (no influence) to 5 (high impact). IFPUG uses 14 factors whereas Mark II adds another five to FPUG's already existing 14, and also allows practitioners to add more to the list.

3. Both methodologies express functional requirements in terms of Base Logical Components (BLC). Mark II uses a single type of BLC and expresses all functional requirements as a catalogue of LT. IFPUG uses five BLC types namely EI, EO, EQ, ILF and EIF.

4. Both focus on logical business requirements.

5. Both define specific rules with regard to the boundary.

6. Both derive base counts, the difference being in how the counts are constructed (refer to the previous sections). In Mark II, the base counts are used directly in the calculation of the functional size index. In IFPUG the base counts are used to determine the complexity (high, medium or low) of each BLC.

7. Weighting - in Mark II, three different weights are used which add up to 2.5, so as to correspond to IFPUG function points. The weighing values have been calculated and are periodically validated, from historical data of numerous projects in a number of large development organisations. These values may be calibrated.

8. Weighting - in IFPUG FPA the weighting system is more complex due to the larger nr. of BLC types. Owing to the classification used, IFPUG's weighting system has a distinct upper limit, which has an impact on the calculation of FP for complex systems.
    - Various differences in detail exist, i.e. the handling of error messages and control information. These differences are not considered vital for our understanding of the concepts and are therefore ignored.

A comparison between Mark II and IFPUG function point counting in particular:

- When the Mark II weighting scale was first developed, the aim was to produce sizes, which were comparable to those of the IFPUG FPA scale.
- As a result of this pegging process, for the range of 200-400 IFPUG UFP (unadjusted function points), the two scales should on average give the same size, once converted. If the IFPUG raw data is available, an additional analysis is made to map the file types of IFPUG to entity types and IFPUG's transaction types to Mark II's LT types. Based on this, the Mark II FP size for the same software can be calculated.
- The FP size relationship for larger applications requires more complex calculations [6].

Examples to illustrate IFPUG FPA and Mark II FPA:

The question discussed here is similar to Question 7 of the Additional Exercises given on page 115 of the textbook.

PARTICULAR DETAIL:

- A program in the College Time-Tabling system produces a report showing students who should be attending each time-tabled teaching activity per semester. The report is produced by an online request.

- The following information is input:
  - course code and
  - semester number.

- The request does not update files.

- The report is printed on the local printer.

- Information printed on the report includes:
  - course code,
  - course description,
  - location,
  - days of the week course is presented,
  - time of course presentation,
  - student numbers, and
  - student names who have enrolled for the course.

- No calculations are made for the report.

- Information is extracted from the following files (data groups):

System	File	Record Types	Data Entity Types
Timetabling system	Time-tabling file. The file is wholly maintained within the boundaries of the Time-Tabling system (of which the report forms part).	The file consists of 2 data groups (record types): one contains location data and the other contains course data.	All the course information is extracted from this file, i.e. location, course code and description, days of the week course is presented, time and duration.
Student system:	The Student detail file is maintained fully by the Student system and is read by the timetabling report.	The file consists of 3 data sets (record types), i.e. general student detail, course detail and financial detail.	For the report, student number, name, course code are extracted from this file.

QUESTIONS:

1. Calculate the UFP using the IFPUG methodology. (Error messages can be ignored for counting purposes.)
2. Convert the FP count into SLOC for a ‘C’ system.
3. Calculate the cost to the University to produce the report, if the programmer who is responsible for producing the report, is charged out at R 500 per day.
4. Calculate the UFP using the Mark II methodology. (Error messages can be ignored for counting purposes.)

SOLUTION:

1. Calculate the UFP using the IFPUG methodology.

STEP	ANALYSIS
Define the boundary	The boundary of the report is the time-tabling system. All files updated within the borders of the time-tabling system will be regarded as Internal Logical Files (ILF); all other files will be regarded as External Interface Files (EIF).
Identify all Data functions	**ILF:** The Time-Tabling file is an ILF. It lies within the boundary of the Time-Tabling system and is maintained only by that system. **EIF:** The Student details file is external to the time-tabling system as it is updated solely by the student system. It is used in a read-only capacity by the report.
Identify all transactional functions	***External Input -*** The request for the report is not an EI. To classify as an EI, the request must update one or more ILF or change the behaviour of the system. ***External Output -*** The report does not classify as an EO transaction. For an EO, it must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILF, or change the behaviour of the system. ***External Inquiry -*** The input and output parts of the report request comply with the **EQ rules.** The primary intent of the input request is to produce a report without updating any ILF. The report printed likewise does not update any ILF nor does it perform any calculation or print any total.

	Trans-action	**Name**	**Record/ File Types**	**Data Types**	**Table**	**Rating**	**Value - Table 5.2**	**IFPUG FP**
Determine the unadjusted function point count	ILF	Time Tabling	2 Record Types	6	5.3	low	7	7
	EIF	Student	3 Record Types	3	5.3	low	5	5
	EQ - Input	Request	1 Time Tabling	2	5.4	low	Greater of two ratings selected: 'average'	4
	EQ - Output	Report	2 Time Tabling & Student	8	5.5	average		
		TOTAL:						**16 IFPUG**

2. Converting the UFP to SLOC for a program written in 'C'.

According to steps 6 & 7, technical complexity factors can be applied to the UFP before converting it into SLOC. In our case, the unadjusted (raw) FP will be used. Depending on the programming language used, the FP can be converted into SLOC. This is a

measure of the size of the program or system. The SLOC represents time and cost. For example, it may be possible for a Programmer to write 100 SLOC per day. Using this, the Project Manager can calculate the cost of new or enhancements to existing systems. According to page 103 of the textbook, 1 IFPUG FP translates to 128 'C' lines of code.

Therefore: 17 FP = 16 x 128 'C' LOC = 2048 'C' LOC

3. Calculate the cost to produce the report.

The SLOC must be converted into code per day. If the Programmer can write 100 tested LOC per day, *then*: 2048 / 100 = 20.48 ≈ 20.5 days to write the report.

Cost to the University: 20.5 days * R 500 per day = R 10 250.00

4. FP calculation according to Mark II Function Point Methodology.

STEP	ANALYSIS			
Define the boundary of the count.	As for IFPUG. Determining the boundary determines the logical transactions to be included in the count, as well as any interfaces across the boundary.			
Identify logical transactions.	A LT consists of an input, output and process part. It is the lowest process supported by the application. In our case, we are dealing with one LT, which will produce a report with data from the Time-Tabling and Student entities. This is essentially a *"LIST"* transaction in Mark II terms.			
Identify and categorise the data and entity types.	Mark II refers to entity types in third normal form. Each occurrence of such an entity type must be counted. The Student file consists of 3 data entity types and the Time-Tabling file of 2 data entity types.			
Identify the input and output components of LT.	The input transaction consists of an inquiry, containing two data element types. The output transaction consists of the report, containing 8 data element types. (The error message is ignored in this Question).			
Calculate the logical transaction size in Mark II function points.	**Type:** Input Process Output **Total**	**Entity Types:** 2 data 3 + 2 8 data	**Mark II Weight:** 0.58 1.66 0.26	**Mark II FP:** 1.16 8.30 2.08 **11.54**

References - Extra reading material:

[1] Burn-Murdoch, M J. *Issues with IFPUG Counting Practices*. Version 4. Software Measurement Services. http://www.gifpa.co.uk

[2] Garmus, D. and Herron, D. 2001. *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison-Wesley.

[3] Garmus, D. *Function Point Counting in a real-time Environment*. David Consulting Group.

[4] Mark II FPA Counting Practices Manual CPM1.3.1, website address: http://www.uksma.co.uk/html/cpm_1.31.html

[5] Rule, G. *A Comparison of the Mark II and IFPUG Variants of Function Point Analysis*. Software Measurement Services, website address: http://www.gifpa.co.uk//library/Papers/Rule/MK2IFPUG.html

[6] Software Measurement Services, *Conversion between IFPUG 4.0 and MkII Function Points,* Version 3, 24 Aug 1999.