Chapter 5: Software effort estimation- part 2

NET481: Project Management

Afnan Albahli

Topics to be covered

- Difficulties of Estimation
- Where are estimates done?
- Problems of over- and under- estimate
- Estimation techniques

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

Albrecht Function Point Analysis

- FP is A top-down method.
- Devised by Allan Albrecht during his work for IBM.
- Why FP?

  To be able to quantify the functional size of programs independently of the programming language used.

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

Albrecht Function Point Analysis (cont'd)

- The basis of FP analysis is that: An Information System consists of five major components or external user types or functions that are of benefit to the user.
- Transaction functions:
  - **External input types**
  - Input transactions that update internal computer files.
  - **External output types**
  - Are transactions where data is output to the user (printed report)
  - **External inquiry types**
  - Are transactions initiated by the user which provide information but not update the internal files.
  - The user inputs some information that directs the system to the details required.

Albrecht Function Point Analysis (cont'd)

- Data functions:
  - **Logical internal file types**
    - The standing files used by the system.
    - File here refers to a group of data items accessed together.
    - It may be made up of one or more record types.
  - **External interface file types**
    - Allow for output and input that may pass to and from other computer systems.
    - Files shared between applications would also be counted here.

Albrecht Function Point Analysis (cont'd)

- The FP approach:

1. Identify each external user type in your application.

2. Determine the complexity of each user type (high, average or low)

3. FP score for of each external user type = Multiply the weight of each complexity by the count of each external user type that has that complexity.

4. **FP count = summation of all the FP scores**.

FP count indicates the size of the information processing.

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

User Type Complexity

- For the original function points defined by Albrecht, the complexity of the components (external user types) was intuitively decided.

- Now there is a group called (IFPUG) international FP user group have put rules governing the complexity and how it is assessed.

- The Albrecht FP is often refereed to as the IFPUG FP method.

IFPUG File Type Complexity

Table 1

External user type	Low	Average	High
External input types	3	4	6
External output types	4	5	7
External inquiry types	3	4	6
Logical internal file types	7	10	15
External interface file types	5	7	10

IFPUG File Type Complexity (cont'd)

Table 2

Number of record types	Number of data types		
	<20	20-50	>50
1	Low	low	Average
2 to 5	Low	Average	High
>5	Average	High	High

- The boundaries shown in this table show how the complexity level for the logical internal files is decided on.
- There are similar tables for external inputs and outputs.
- Record Type is also called Record Element Type (RET)
- Data Type is also called Data Element Type (DET

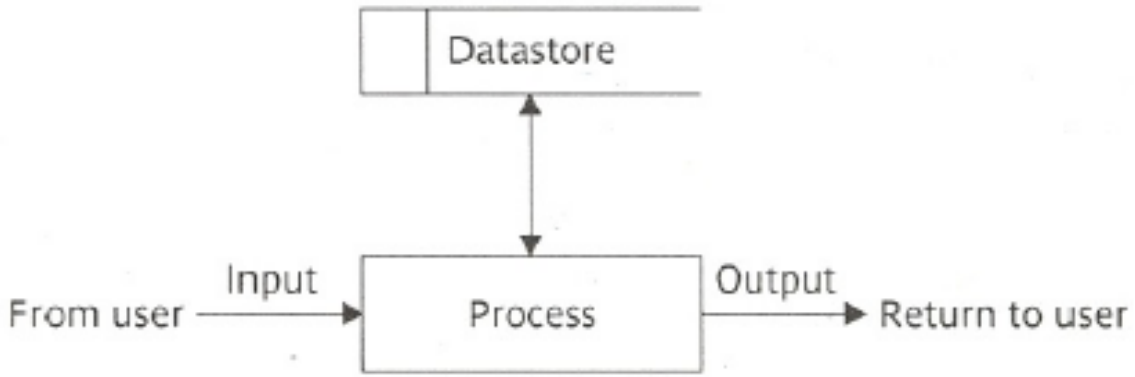
Function Points Mark II

- Developed by Charles Symons in 1991.
- It is not a replacement to the Albrecht method (the IFPUG method)
- FP Mark II as Albrecht FPs measures the information processing size in FPs.

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

Function Points Mark II (cont'd)

- The idea of FP Mark II: an information system contains transactions which have the basic structure shown below:



Function Points Mark II (cont'd)

- FP = Wi * (number of input data element types) +

  We * (number of entity types referenced) +

  Wo * (number of output data element types)

- Wi, We, Wo are weightings derived by asking developers the proportions of effort spent in previous projects developing the code dealing with:

✓ Inputs

✓ Accessing and modifying stored data

✓ Processing outputs

Function Points Mark II (cont'd)

- The proportions of effort are then normalized into ratios or weightings, which add up to 2.5.

- 2.5 was adopted to produce FP counts similar to the Albrecht equivalents.

- Industry averages for the weights:

Wi = 0.58 , We= 1.66 , Wo = 0.26 they add up to 2.5

Example

A cash receipt transaction in the IOE maintenance accounts subsystem accesses two entity types – INVOICE and CASH-RECEIPT.

The data inputs are:

Invoice number

Date received

Cash received

If an INVOICE record is not found for the invoice number then an error message is issued. If the invoice number is found then a CASH-RECEIPT record is created. The error message is the only output of the transaction. The unadjusted function points, using the industry average weightings, for this transaction would therefore be:

Answer

- ✓ FP = Wi * (number of input data element types) +
  We * (number of entity types referenced) +
  Wo * (number of output data element types)

- ✓ Wi = 0.58 , We= 1.66 , Wo = 0.26
- ✓ number of input data
  eleme
  nt types = 3 (Invoice number, Date received, Cash received)
- ✓
  num
  ber of entity types referenced = 2 (Invoice and Cash- receipt)
- ✓ number of output data element types = 1(error message)
- ✓ FP = (0.58*3) + (1.66*2) + (0.26*1) = 5.32

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

COSMIC Full Function points

- COSMIC FFPs stands for Common Software Measurement Consortium Full Function Points.

- This approach is developed to measure the sizes of real-time or embedded systems.

- In COSMIC method: the system architecture is decomposed into a hierarchy of software layers.

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

COSMIC Full Function points (cont'd)

They define 4 data groups that a software component can deal with:

- **Entries (E).** effected by sub-processes that moves the data group into the SW component in question from a user outside its boundary.
- **Exits (X).** effected by sub-processes that moves the data group from the SW component into a user outside its boundary.
- **Reads (R).** data movements that move data groups from a persistent storage (DB) to the SW component.
- **Writes (W).** data movements that move data groups from the SW component to a persistent storage

COSMIC Full Function points (cont'd)

- The overall FFP is derived by simply summing the counts of the four groups all together.

- The method doesn't take account of any processing of the data groups once they are moved into the software component.

- It is not recommended for systems that include complex mathematical algorithms.

COCOMO II

- It is a parametric productivity model.
- It is developed by Barry Boehm in the late 1970s.
- COCOMO is short for COnstructive COst Model.
- It refers to a group of models.
- The basic model was built around the following equation:
  - Effort= c(size)k
  - The effort is measured in person-months (pm), consisting of units of 152 working hours.
  - The size is measured in (Kdsi) thousands of delivered source code of instructions.
  - c and k are constants.

COCOMO II (cont'd)

- The first step is to derive an estimate the system size in terms of kdsi.
- C and k constants values depend on classifying the system in

Boehm's terms as either:

- ✓ Organic mode or
- ✓ Embedded mode or
- ✓ Semi-detached mode.

SPM (5e) Software effort estimation©
The McGraw-Hill Companies, 2009

COCOMO II (cont'd)

- Organic mode.
  - Small team,
  - Small system,
  - Interface requirements flexible,
  - In-house software development.
- Examples:

Systems such as payroll, inventory.

COCOMO II (cont'd)

- Embedded mode.
  - Product has to operate within very tight constraints,
  - the project team is large,
  - development environment consists of many complex interfaces,
  - Changes are very costly.
- Examples:

Real-time systems such as those for air traffic control, ATMs,

or weapon systems.

COCOMO II (cont'd)

- Semi-detached mode.
  - Combined elements from the two above modes or characteristics that come in between.
- Examples:

Systems such as compilers, database systems, and editors.

COCOMO II (cont'd)

- c and k values

System type	c	k
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

COCOMO II (cont'd)

- COCOMO II takes into account that there is a wider range of process models in use than before.

- COCOMO II is designed to accommodate the fact that estimates will be needed at different stages of the system life cycle.

- COCOMO II has models for three different stages:
  - Application composition.
  - Early design.
  - Post Architecture.