- Comments and blank lines should not be counted.

Function Point Metric

- Overcomes some of the shortcomings of the LOC metric
- A set of related inputs is counted as one input.

Big Questions

1.Interpret about the Estimation of Software and Software Effort Estimation Technique.

Estimation of Software

- ✓ Software projects are notorius for going past their deadline, going over budget, or both.
- ✓ The problem lies in the estimation of the amount of effort required for the development of a project.
- ✓ The cost estimation is usually dependent upon the size estimate of the project, which may use lines of code or function points as metrics.
- ✓ There are several different techniques for performing software cost estimation, including expert judgement and algorithmic models. Estimation by expert judgement is a common way of estimating the effort required for a project. Unfortunately, this method of estimation does not emphasize re-estimation during the project life cycle, which is an important part of project tracking, because it allows the estimates to be improved during the project life cycle.
- ✓ The quality of a cost estimation model is not so much attributed to the initial estimate, but rather the speed at which the estimates converges to the actual cost of the project.
- ✓ COCOMO is a popular algorithmic model for cost estimation whose cost factors can be tailored to the individual development environment, which is important for the accuracy of the cost estimates. More than one method of cost estimation should be done so that there is some comparison available for the estimates.
- ✓ This is especially important for unique projects. Cost estimation must be done more diligently throughout the project life cycle so that in the future there are fewer surprises and unforeseen delays in the release of a product.

Basic Steps in Software Effort Estimation

- ✓ Identify project objectives and requirements
- ✓ Plan the activities
- ✓ Estimate product size and complexity
- ✓ Estimate effort, cost and resources
- ✓ Develop projected schedule
- ✓ Compare and iterate estimates
- ✓ Follow up

Basic Algorithmic Form

Effort = constant + coefficient*(size metric) + coefficient*(cost driver 1) + coefficient*(cost driver 2) + coefficient*(cost driver 3) + …

size metric lines of code

‘new’ versus ‘old’ lines of code

function points

SLOC as an Estimation Tool

- ✓ Why used ?
  - o early systems emphasis on coding
- ✓ Criticisms
  - o cross-language inconsistencies
  - o within language counting variations
  - o change in program structure can affect count
  - o stimulates programmers to write lots of code
  - o system-oriented, not user-oriented

2. (i) Briefly demonstrate the effort and cost estimation techniques

(ii) Analyze the importance of COCOMO model for software estimation

(i) Effort and Cost estimation techniques

Estimating

- ✓ The process of forecasting or approximating the time and cost of completing project deliverables.
- ✓ The task of balancing the expectations of stakeholders and the need for control while the project is implemented

Types of Estimates

- ✓ Top-down (macro) estimates : analogy, group consensus, or mathematical relationships, derived from experience to estimate project duration and total cost. Could be made by a manager with no direct experience of the processes to complete the project.
- ✓ Bottom-up (micro) estimates: estimates of elements of the work breakdown structure, require more effort to develop & rely upon those who understand the work to estimate specific work activities

Macro (Top-down) Approaches

- ❖ Consensus methods
- ❖ Ratio methods
- ❖ Apportion method
- ❖ Function point methods for software and system projects
- ❖ Learning curves

Micro (Bottom-up) Approaches

- ❖ Template method
- ❖ Parametric Procedures Applied to Specific Tasks
- ❖ Detailed Estimates for the WBS Work Packages
- ❖ Phase Estimating: A Hybrid

(ii) COCOMO Model

- ❖ COCOMO (COnstructive COst MOdel) proposed by Boehm.
- ❖ Divides software product developments into 3 categories :
  - o Organic
  - o Semidetached
  - o Embedded

COCOMO Product classes

- ❖ Roughly correspond to:
- ❖ Application, utility and system programs respectively.
- ❖ Data processing and scientific programs are considered to be application programs.
- ❖ Compilers, linkers, editors, etc., are utility programs.
- ❖ Operating systems and real-time system programs, etc. are system programs.

Software cost estimation is done through three stages :

- o Basic COCOMO,
- o Intermediate COCOMO,
- o Complete COCOMO.

Basic COCOMO Model

- ❖ Gives only an approximate estimation :
  - o Effort = a1 (KLOC)a2
  - o Tdev = b1 (Effort)b2
- ❖ KLOC is the estimated kilo lines of source code,
- ❖ a1,a2,b1,b2 are constants for different categories of software products,
- ❖ Tdev is the estimated time to develop the software in months,
- ❖ Effort estimation is obtained in terms of person months (PMs).

Intermediate COCOMO

- ❖ Basic COCOMO model assumes
  - o effort and development time depend on product size alone
- ❖ However, several parameters affect effort and development time:
  - o Reliability requirements
  - o Availability of CASE tools and modern facilities to the developers
  - o Size of data to be handled
- ❖ For accurate estimation,
  - o the effect of all relevant parameters must be considered:
  - o Intermediate COCOMO model recognizes this fact:

- ❖ Refines the initial estimate obtained by the basic COCOMO by using a set of 15 cost drivers (multipliers).

Complete COCOMO

- ❖ Cost of each sub-system is estimated separately.
- ❖ Costs of the sub-systems are added to obtain total cost.
- ❖ Reduces the margin of error in the final estimate.

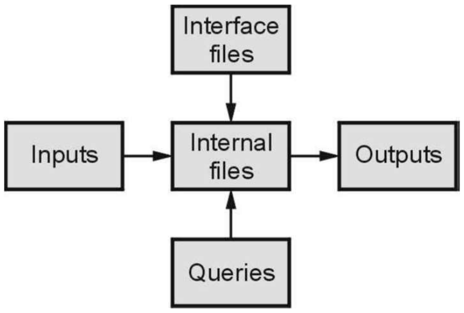
Complete COCOMO Example

- ❖ A Management Information System (MIS) for an organization having offices at several places across the country :
  - o Database part (semi-detached)
  - o Graphical User Interface (GUI) part (organic)
  - o Communication part (embedded)
- ❖ Costs of the components are estimated separately :
  - o summed up to give the overall cost of the system

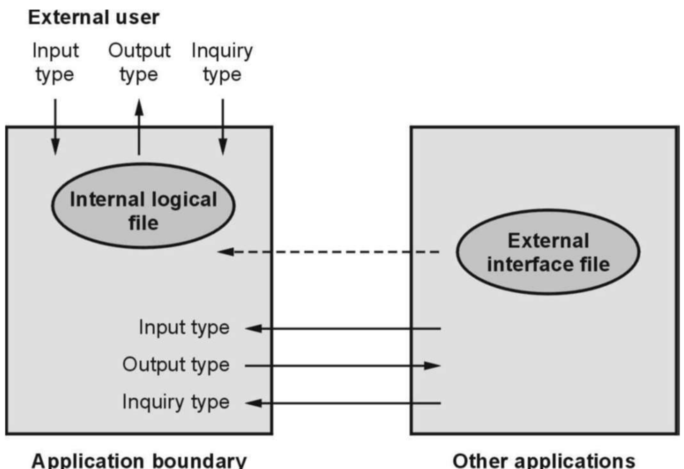
3. How do you find the cosmic full function point measurement method associates the functional user requirements for each piece with a specific layer as each layer possesses an intrinsic boundary for which specific users are identified?

COSMIC Full function point

COSMIC function points are a unit of measure of software functional size. There are other ways to assess software size, but COSMIC Sizing is the most universal, meaningful and useful approach. The functional size is consistent regardless of the technology used to build it. The size can be estimated, or if all requirements are available, measured. Early estimation is very useful for planning and managing software endeavours (projects or product management). The process of measuring software size is called functional size measurement (FSM). COSMIC functional size measurement is applicable to business software, real-time software and infrastructure software at any level of decomposition (from a whole software system down to a single re-usable component or a user story). The functional size is independent of the technology or processes used to develop the system. COSMIC is an ISO standard. It is a refined improvement over its predecessors (IFPUG and Mark II FP). The unit of size is the COSMIC Function Point or CFP.



Functionality Types


External user
Input type
Output type
Inquiry type
Internal logical file
External interface file
Input type
Output type
Inquiry type
Application boundary
Other applications